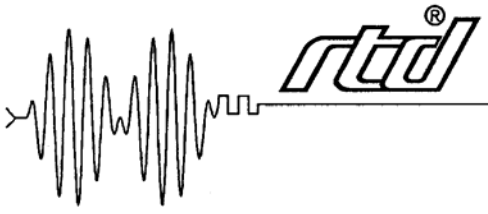
EFM104HR
Data/Fax Modem Module

User's Manual



Real Time Devices, Inc.

"Accessing the Analog World"®

ISO9001 and AS9100 Certified

BDM-610020041 Rev A

EFM104HR
56 kbaud Data/Fax Modem Module
User's Manual

RTD Embedded Technologies
103 Innovation Blvd.
State College, PA 16803
USA

Phone: (814) 234-8087

Email
sales@rtd.com
techsupport@rtd.com

Websites
http://www.rtd.com/

WARNING

LIFE SUPPORT APPLICATIONS

This product is not designed for use in life support appliances, devices or systems where malfunctioning of these products can reasonably be expected to result in personal injury. RTD customers using or selling this product for use in these applications do so at their own risk and fully agree to indemnify RTD for any damages resulting from such improper use or sale.

Revision History

13/08/2001	HW Release 1.0, Preliminary version, released
Rev A	Updated names and added BDM number

Notice: Information in this manual may change without prior notice.

Published by:
RTD Embedded Technologies, Inc.
103 Innovation Boulevard
State College, PA 16803
Copyright 2005 by RTD Embedded Technologies, Inc.
All rights reserved
The RTD Embedded Technologies Logo is a registered trademark of RTD Embedded Technologies. dspModule, cpuModule, and utilityModule are trademarks of RTD Embedded Technologies. PC/104, PC/104-Plus, and PCI-104 are registered trademark of PC/104 Consortium.

Table of Contents

List of illustrations and tables 5

Chapter 1 Introduction 6
Features 6
Data/Fax modem 6
16C550 compatible UART 7
I/O interfaces 7
Mechanical description 7
Connector description 7
What comes with your board 7
Using this manual 8
When you need help 8

Chapter 2 Board settings 9
Factory configured jumper settings 10
Base address jumpers 11
Host interrupt 13

Chapter 3 Board installation 14
Board installation 14
General purpose digital I/O 15

Chapter 4 Hardware description 17
The 56K flex modem module 18
Phone line connection 18
Status LED's 18
UART channel 19
Digital I/O 19

Chapter 5 Board operation and programming 20
Defining the memory map 20
BASE+400h Digital I/O 21
BASE+402h EFM104HR status register 21
Interrupts 22

Chapter 6 EFM104HR Specifications 28

Chapter 7 Limited Warranty 29

List of illustrations and tables

Fig. 2-1 EFM104HR Board layout showing jumper locations

Fig. 2-2 Base address jumpers illustrating address 3F8h

Fig. 2-3 Interrupt jumpers from left to right: IRQ 2, 5, 6,7,10,11,12,15 and G

Fig. 3-1 EFM104HR integrated in a RTD PC/104 cpuModule stack together with CMM series cpuModule and dataModule

Fig. 3-2 Digital I/O connector layout of the EFM104HR

Fig. 4-1 Block diagram of the EFM104HR

Table 2-1 Factory configured jumper settings

Table 2-2 Base address jumper settings EFM104HR

Table 3-1 Pin outs of the EFM104HR digital I/O interface connector

Table 5-1 General I/O map of the EFM104HR

Chapter 1 - INTRODUCTION

This user's manual describes the operation of the RTD EFM104HR embedded modem module designed for industrial, telemetry and security applications.

Features

Some of the key features of the EFM104HR include:

- Low power MT5634SMI-ITP 56K Data/Fax modem
- UL 1950, FCC part 68, CS03 and EN60950 approved
- 14.4K Class 1 and 2 fax services
- Industry standard AT-style commands
- 16C550 UART interface to host computer
- Supports COM1,COM2,COM3,COM4 or COMx
- Available IRQ's 2,5,6,7,10,11,12,14,15
- Status LED's for DTR,CTS, TXD and RXD
- 16 TTL I/O's 8 outputs, 8 inputs with 10K pull down
- +5V only operation, 1W power consumption
- Wide operating temperature range –40 to + 85C
- Fully PC/104 compliant, IDAN versions available

The following paragraphs briefly describe the major features of the EFM104HR. A more detailed discussion is included in Chapter 4 (Hardware description). The board's installation is described in Chapter 2 (Board Installation).

Data/Fax modem

The Real Time Devices EFM104HR embedded data and fax modem provides a direct and reliable connection to proprietary or public wired telephone systems for data and/or data communication. The MT ModemModule complies with telecom requirements in the areas of US, Canada and the EU.

The EFM104HR uses the Multi-Tech MT5635SMI-ITP industrial temperature range modem module. This includes a Lucent Venus controller + DSP and the Lucent 1034CSP codec. It also includes a 4M Flash and 32Kx16 SRAM for V.90/K56flex modem operation and V.17 Class 1 and Class 2 Fax.

16C550 compatible UART

Communication to the ModemModule is performed through a standard UART channel. This onboard serial port leaves the other system serial ports free for the user. All operating systems will recognize and support this 16C550 standard UART, and therefore no special communication drivers are needed to receive data from your modem. Commercial K56flex modem drivers will work correctly. The address and interrupt of your serial channel can be changed with the onboard jumpers.

I/O interfaces

The EFM104HR can be controlled and monitored from the software through two dedicated I/O registers. A special I/O connector is available for the user to connect to the general-purpose TTL level digital I/O. The control registers are located in the I/O area of BASE+400h.

Mechanical description

The EFM104HR is designed on a PC/104 form factor. An easy mechanical interface to both PC/104 and RTD IDAN systems can be achieved. Stack your EFM104HR directly on a PC/104 compatible CPU module using the onboard mounting holes and standoffs.

Connector description

The Line interface uses a RJ11 standard modem jack interface. Connect your phone cable directly to this connector, or use a short cable inside your enclosure to connect to a feed through connector to allow connection of the antenna to the wall of your enclosure. All general digital I/O connections are made using header type terminals.

What comes with your board

Your EFM104HR package contains the following items:

- EFM104HR board
- Companion CD with Manual and Drivers

Note: Device drivers and example software available on the internet

If any item is missing or damaged, please contact RTD Embedded Technologies.

Using this manual

This manual is intended to help you install your new EFM104HR module and get it working quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in the most demanding applications.

When you need help

This manual and all the example programs will provide you with enough information to fully utilize all the features on this board. If you have any problems installing or using this board, contact our Technical support department at techsupport@rtd.com, or call (814) 234-8087. When sending us an Email request, please include the following information: Your company's name and address, your name, your telephone number, and a brief description of the problem.

Chapter 2 - BOARD SETTINGS

The EFM104HR board has jumper settings, which can be changed to suit your application and host computer configuration. The factory settings are listed and shown in the diagram at the beginning of this chapter. Make sure you completely study and understand this chapter before making changes to these settings.

Factory-Configured Jumper Settings

Table 2-1 below illustrates the factory jumper setting for the EFM104HR. Figure 2-1 shows the board layout of the EFM104HR and the locations of the jumpers. The following paragraphs explain how to change the factory jumper settings to suit your specific application.

Table 2-1 Factory configured jumper settings (Please see figure 2-1 below for more detailed locations)

JUMPER NAME	DESCRIPTION	NUMBER OF JUMPERS	FACTORY SETTING
BASE	Base Address	6	2E8 / 6E8
IRQ	Host interrupt	11+1	5, G – jumper closed

Fig. 2-1 EFM104HR Board layout showing jumper locations

Base address jumpers (Factory setting: 2E8h, 6E8h)

The EFM104HR is I/O mapped into the memory space of your host XT/AT. The board occupies a consecutive memory window of 8 bytes starting from the base address for UART communication and 4 consecutive bytes starting from BASE+400h for the board control and status registers. As an example if your base address is set to be 2E8h for the serial port, the onboard control registers will start from 6E8h.

The most common cause of failure when you are first setting up your module is address contention: some of your computers I/O space is already occupied by other devices and memory resident programs. When the EFM104HR attempts to use its own reserved memory addresses (which are being already used by another peripheral device) erratic performance can occur and the data read from the board may be corrupted.

To avoid this problem make sure you set up the base address by using the six jumpers on the right side of the board, this allows you to choose from a number of different addresses in your host computer's I/O map. Should the factory installed setting of 38fh be incompatible to your system configuration, you may change this setting to another using the options illustrated in Table 2-2 (overleaf). The table shows the jumper settings and their corresponding values in hexadecimal form. Ensure that you verify the correct location of the base address jumpers. When the jumper is removed it corresponds to a logical "0", connecting the jumper to a "1". When you set the base address of the module, record the setting inside the back cover of this manual.

EFM104HR Base address configuration

BASE	A8	A7	A6	A5	A4	A3	BASE	A8	A7	A6	A5	A4	A3
200	0	0	0	0	0	0	300	1	0	0	0	0	0
208	0	0	0	0	0	1	308	1	0	0	0	0	1
210	0	0	0	0	1	0	310	1	0	0	0	1	0
218	0	0	0	0	1	1	318	1	0	0	0	1	1
220	0	0	0	1	0	0	320	1	0	0	1	0	0
228	0	0	0	1	0	1	328	1	0	0	1	0	1
230	0	0	0	1	1	0	330	1	0	0	1	1	0
238	0	0	0	1	1	1	338	1	0	0	1	1	1
240	0	0	1	0	0	0	340	1	0	1	0	0	0
248	0	0	1	0	0	1	348	1	0	1	0	0	1
250	0	0	1	0	1	0	350	1	0	1	0	1	0
258	0	0	1	0	1	1	358	1	0	1	0	1	1
260	0	0	1	1	0	0	360	1	0	1	1	0	0
268	0	0	1	1	0	1	368	1	0	1	1	0	1
270	0	0	1	1	1	0	370	1	0	1	1	1	0
278	0	0	1	1	1	1	378	1	0	1	1	1	1
280	0	1	0	0	0	0	380	1	1	0	0	0	0
288	0	1	0	0	0	1	388	1	1	0	0	0	1
290	0	1	0	0	1	0	390	1	1	0	0	1	0
298	0	1	0	0	1	1	398	1	1	0	0	1	1
2A0	0	1	0	1	0	0	3A0	1	1	0	1	0	0
2A8	0	1	0	1	0	1	3A8	1	1	0	1	0	1
2B0	0	1	0	1	1	0	3B0	1	1	0	1	1	0
2B8	0	1	0	1	1	1	3B8	1	1	0	1	1	1
2C0	0	1	1	0	0	0	3C0	1	1	1	0	0	0
2C8	0	1	1	0	0	1	3C8	1	1	1	0	0	1
2D0	0	1	1	0	1	0	3D0	1	1	1	0	1	0
2D8	0	1	1	0	1	1	3D8	1	1	1	0	1	1
2E0	0	1	1	1	0	0	3E0	1	1	1	1	0	0
2E8	0	1	1	1	0	1	3E8	1	1	1	1	0	1
2F0	0	1	1	1	1	0	3F0	1	1	1	1	1	0
2F8	0	1	1	1	1	1	3F8	1	1	1	1	1	1

0 = JUMPER OFF **1 = JUMPER CLOSED**

Table 2-2 Base address jumper settings for the EFM104HR

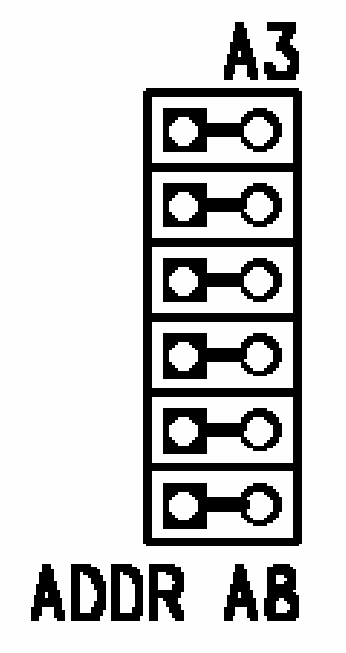


Fig. 2-2 Base address jumpers illustrating address 3F8h, A8 is to the bottom, A3 is located to the top of the jumper block

Host interrupt (Factory setting: IRQ5, G closed)

The header connector, shown in Figure 2-3 below, lets you connect the onboard control logic interrupt outputs to one of the interrupt channels available on the host computer XT/AT bus.

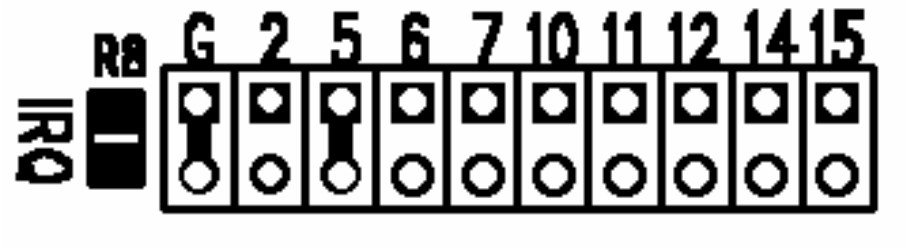


Fig. 2-3 Interrupt jumpers from left to right IRQ2,5,6,7,10,11,12,14,15 and G

Note: The EFM104HR hardware supports interrupt sharing! Jumper G must be closed on one module per used interrupt. For example if two boards share interrupt number 7 only one board may have the G jumper closed. The G jumper connects a 1KOhm resistor to ground while the shared interrupts are 3-stated pulling the line to an inactive level.

Chapter 3 BOARD INSTALLATION

The EFM104HR GSM modem module is designed to directly mount on top or under your RTD PC/104 cpuModule stack. This chapter tells you step-by-step how to install your EFM104HR into your system.

Board installation

Keep your board in its antistatic bag until you are ready to install it to your system! When removing it from the bag, hold the board at the edges and do not touch the components or connectors. Please handle the board in an antistatic environment and use a **grounded** workbench for testing and handling of your hardware. Before installing the board in your computer, check the power cabling. Failure to do so may cause the power supply unit to malfunction or even cause permanent damage.

General installation guidelines:

- Touch the grounded metal housing of your computer to discharge any antistatic buildup and then remove the board from its antistatic bag.
- Hold the board by the edges and install it in an enclosure or place it on the table on an antistatic surface
- Install your board in your system, and wire the power supply correctly. Failure to do so may cause the power supply unit to malfunction or even cause permanent damage to the device.
- Check all wiring connections once and then once more again.
- Connect the phone line jack to the RJ11 connector on the EFM104HR modem.
- Apply power to your system.

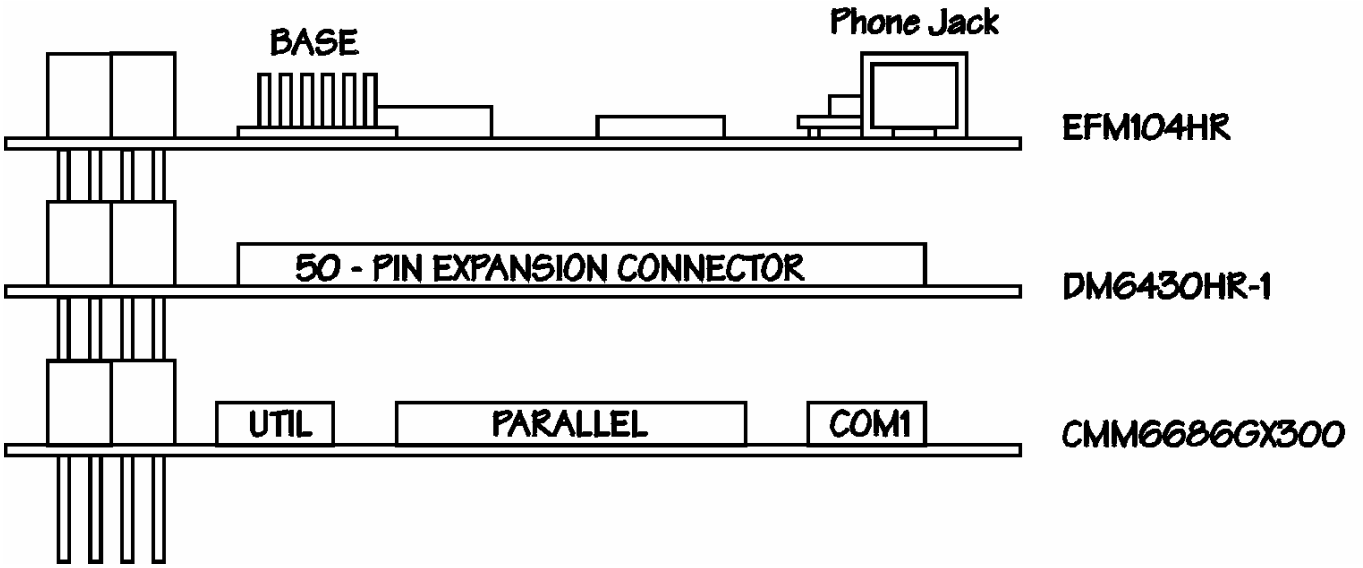


Fig. 3-1 EFM104HR integrated in a RTD PC/104 cpuModule stack together with a CMM series cpuModule and dataModule

General Purpose Digital I/O connector

The Table 3-1 below shows the pin outs of the EFM104HR digital I/O interface. The signals in this connector can be used as general purpose TTL level I/O lines to interface to LCD displays, LED's, push buttons or relays. Note that Figure 3-2 shows two connectors together. The connector J2 carries all inputs and J3 carries all the outputs.

PIN J3	**Description**	**PIN J3**	**Description**
1	+5V	2	Out0
3	Out1	4	Out2
5	Out3	6	Out4
7	Out5	8	Out6
9	Out7	10	GND
PIN J2	**Description**	**PIN J26**	**Description**
1	+5V	2	In0
3	In1	4	In2
5	In3	6	In4
7	In5	8	In6
9	In7	10	GND

Table 3-1 Pin outs of the EFM104HR digital I/O interface connector

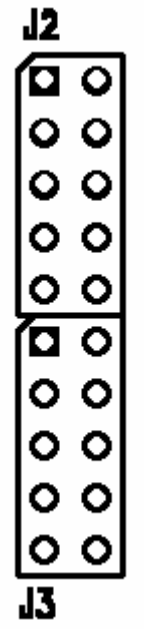


Fig 3-2 Digital I/O connector layout of the EFM104HR

Chapter 4 - HARDWARE DESCRIPTION

This chapter describes the major hardware building blocks of the EFM104HR:

- The 56K flex modem module
- Phone line connection
- Status LED's
- UART channel
- Digital I/O

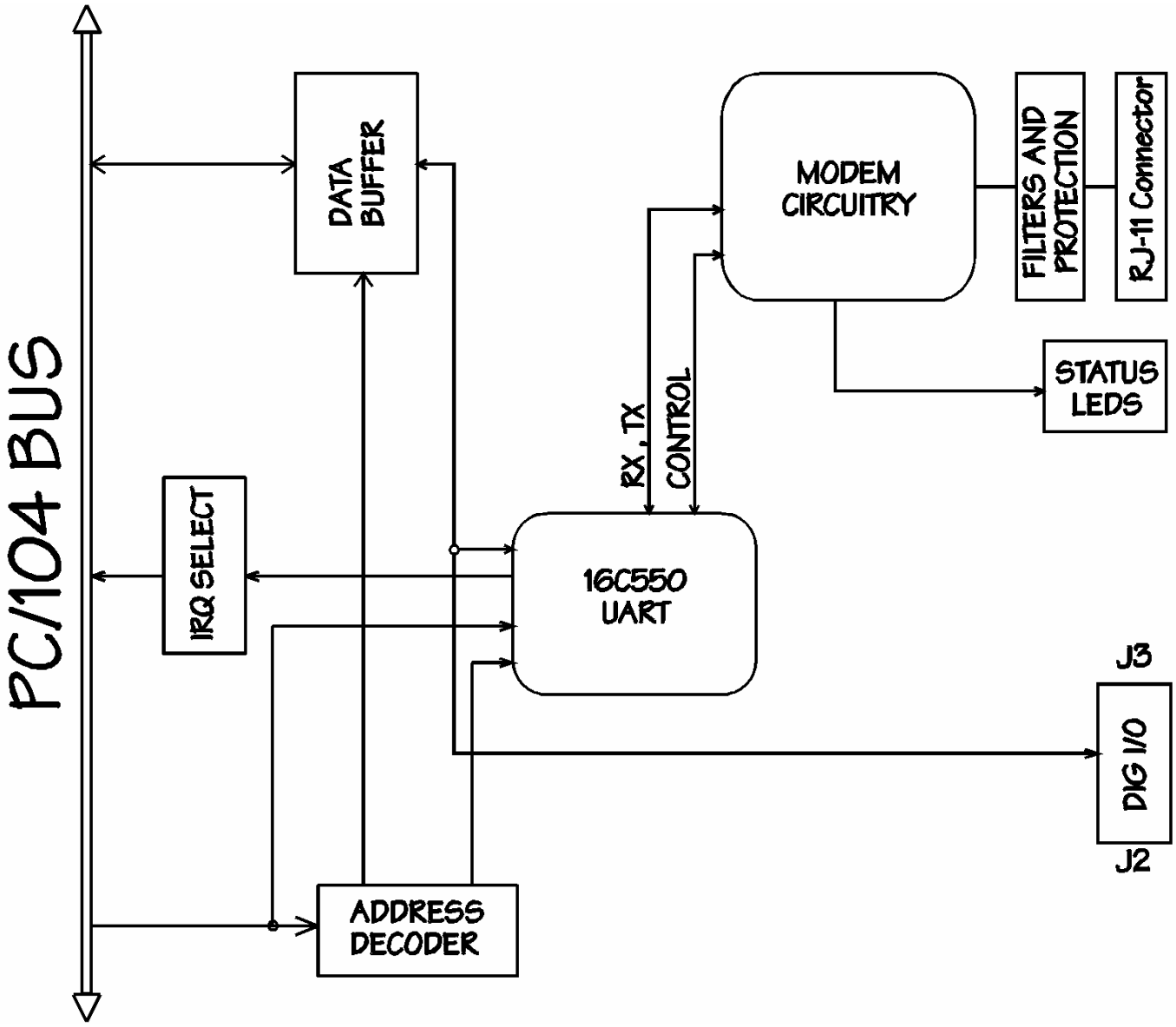


Fig. 4-1 Block diagram of the EFM104HR

The 56K flex modem module

The EFM104HR modem is built around the MultiTech industrial 56 Kbaud modem module. It is designed for applications such as telemetry, telematics or communication and for integration in stationary telephone systems in the EU, the US and Canada.

The EFM104HR is capable of powerful communication with a data rate of 56Kbaud. It is capable of FAX communication at 14,4kbaud. The EFM104HR uses standard modem AT commands with a special extension instruction set for modem specific functions. A complete description on these AT instructions is available in the component specific documentation of the MT5634SMI modem unit.

Phone line connection

This section discussed hardware issues related to the phone line connection, protection and filtering. Surface mount EMC-filtering ferrites are used on the T&R to reduce emissions on the RJ11 cable. 220pF capacitors are also used to reduce common mode emissions that may be present in certain systems. On the solder side is a blob that must be closed if the mounting hole next to the high voltage capacitors is not connected to the chassis ground of your computer.

NOTE: Even if these precautions on the board and the modem module are followed, there are no guarantees that a particular installation or system will comply with all the necessary regulatory requirements. It is imperative that specific systems are evaluated by a qualified or recognized agency in you country.

Status LED's

Four LED's are used to indicate communication activity. Two green LED's indicate TXD and RXD line activity while two red LED's show connection status with DTR and DCD signals.

UART channel

Modem data is sent and received through a standard 16C550 compatible UART. All today's operating systems will recognize and support this serial communication device. The EFM104HR uses its own onboard serial port and will not reserve serial port resources from the system. The I/O base address and interrupt for this serial port can be flexibly set as has been described in previous chapters of this manual. This user's manual will not wade into details of serial port programming. This information is commonly available today. You can use any communication software package or terminal program to connect to your EFM104HR UART. Just make sure you set up the I/O and IRQ right. The UART on the board is specified for full operation from –40 to +85C. The oscillator frequency is set to be 1.8432MHz. Note that the UART interrupt can be disabled or enabled from software by writing to bit 01 in address 0x402. After power-up the interrupt is enabled.

Digital I/O

For general-purpose digital I/O interfacing a 16-bit digital I/O port is provided. This port includes 8 TTL-level digital outputs that are automatically cleared (to 0) after system reset. Also are included 8 digital inputs with 10K Ohm pull-down resistors. These I/O's are located on the left side of the board. These I/O's are ideal to be used to interface to LCD displays, LED's push buttons or other low power controls.

Chapter 5 BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your EFM104HR. It provides a general description of the I/O map. Detailed serial port programming tips are not within the scope of this manual.

Defining the Memory Map

The memory map of the EFM104HR occupies eight bytes of host PC I/O space. This window is freely selectable by the user as described in *Chapter 2, Table 2-2.* After setting the base address you have access to the internal resources of the EFM104HR control logic. These resources are not described in detail, since they are mapped as a standard PC serial port. For more details on the EXAR ST16C550IJ44 UART chip programming please download the component specific data sheet from the manufacturer's website: http://www.exar.com/products/st16c550.html

ADDR (hex)	REGISTER	DIR	COMMENTS
BASE	TXD	O	Only if control reg. Bit 7=0
	RXD	I	Only if control reg. Bit 7=0
	BAUD div. Low		Only if control reg. Bit 7=1
BASE+1	BAUD div. High		Only if control reg. Bit 7=1
	IRQ enable		Only if control reg. Bit 7=0
BASE+2	IRQ ID		
BASE+3	Line control		
BASE+4	Modem control		
BASE+5	Line status		
BASE+6	Modem status		
BASE+400	Digital I/O	I/O	Digital I/O ports
BASE+402	EFM104HR status	I/O	Configuration registers
BASE+403	EFM104HR control	I/O	Reserved for future use

Table 5-1 General I/O map of the EFM104HR, BASE = Base Address

BASE+400 Digital I/O (R/W)

This address is used to interface to the digital I/O port of the EFM104HR; writing to this address will transfer the data out of the output port, while reading from this address will return the data from the digital inputs.

BASE+402 EFM104HR Status (R/W, 0x00 after reset)

Write

Bit 0	/EN_RST	0 - host reset will clear digital outputs; 1 - disabled
Bit 1	/EN_INT	0 – UART interrupt enabled; 1 - disabled
Bit 2	RESERVED	
Bit 3	RESERVED	

Read

Bit 0	/EN_RST	state
Bit 1	/EN_INT	state
Bit 2	RESERVED	
Bit 3	RESERVED	

BASE+403 EFM104HR Control (R/W, 0x00 after reset)

Write

Bit 0	RESERVED
Bit 1	RESERVED
Bit 2	RESERVED
Bit 3	RESERVED

Read

Bit 0	RESERVED
Bit 1	RESERVED
Bit 2	RESERVED
Bit 3	RESERVED

INTERRUPTS

What is an interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are a very flexible way of dealing with asynchronous events. Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time to do nothing whilst waiting for a keystroke to occur. Thus the interrupt scheme is used and the processor proceeds with other tasks. When a keystroke finally occurs, the keyboard then 'interrupts' the processor so that it can get the keyboard data. It then places it into the memory, and then returns to what it was doing before the interrupt occurred. Other common devices that use interrupts are A/D boards, network boards, other used serial ports etc.

Interrupt request lines

To allow different peripheral devices to generate interrupts on the same computer, the PC AT bus has interrupt request channels (IRQ's). A rising edge transition on one of these lines will be latched into the interrupt controller. The interrupt controller checks to see if the interrupts are to be acknowledged from that IRQ and, if another interrupt is being processed, it decides if the new request should supersede the one in progress or if it has to wait until the one in progress has been completed. The priority level of the interrupt is determined by the number of the IRQ as follows; IRQ0 has the highest priority whilst IRQ15 has the lowest. Many of the IRQ's are already used by the standard system resources, IRQ0 is dedicated to the internal timer, IRQ1 is dedicated to the keyboard input, IRQ3 for the serial port COM2, and IRQ4 for the serial port COM1. Often interrupts 2,5,7,10,11 and 15 are free for the user.

8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in a PC is the 8259 Interrupt Controller. To use interrupts you will need to know how to read and set the 8259's internal interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to acknowledge the 8259 interrupt controller.

Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of the interrupt line. If a bit is set (equal to 1), then the corresponding IRQ is masked, and it will not generate an interrupt. If a bit is cleared (equal to 0), then the corresponding IRQ is not masked, and it can then generate an interrupt. The interrupt mask register is programmed through **port 21h**.

End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 Interrupt Controller must be acknowledged by writing the value 20h to port 20h.

What exactly happens when an interrupt occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to correctly write interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the EFM104HR), the interrupt controller checks to see if interrupts are enabled for that IRQ. It then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed onto the system stack, and a new set if CS and IP are loaded from the lowest 1024 bytes of memory.

This table is referred to as the interrupt vector table and each entry to this table is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor starts to execute code from the new Code Segment (CS) and from the new Instruction Pointer (IP). When the interrupt routine is completed, the old CS and IP are popped from the system stack and the program execution continues from the point where interruption occurred.

Using Interrupts in your Program

Adding interrupt support to your program is not as difficult as it may seem especially when programming under DOS. The following discussion will cover programming under DOS. Note that even the smallest mistake in your interrupt program may cause the computer to hang up and will only restart after a reboot. This can be frustrating and time-consuming.

Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write an interrupt service routine (ISR). This is the routine that will be executed automatically each time an interrupt request occurs for the specified IRQ. An ISR is different from other sub-routines or procedures. First on entrance the processor registers must be pushed onto the stack before anything else! Second, just before exiting the routine, you must clear the interrupt on the EFM104HR by writing to the Status register, and write the EOI command to the interrupt controller. Finally, when exiting the interrupt routine the processor registers must be popped from the system stack and you must execute the IRET assembly instruction. This instruction pops the CS, IP and processor flags from the system stack. These were pushed onto the stack when entering the ISR.

Most compilers allow you to identify a function as an interrupt type and will automatically add these instructions to your ISR with one exception: most compilers do not automatically add the EOI command to the function, you must do it yourself. Other than this and a few exceptions discussed below, you can write your ISR as any code routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend you stick to the basics; just something that enables you to verify you have entered the ISR and executed it successfully. For example: set a flag in your ISR and in your main program check for the flag.

Note: If you choose to write your ISR in in-line Assembly, you must push and pop registers correctly and exit the routine with the IRET instruction instead of the RET instruction.

There are a few precautions you must consider when writing ISRs. The most important is: **do not use any DOS functions or functions that call DOS functions from an interrupt routine.** DOS is not re-entrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about using interrupts? Consider then the following situation in your program: If DOS function X is being executed when an interrupt occurs and the interrupt

routine makes a call to the same DOS function X, then function X is essentially being called while active. Such cases will cause the computer to crash. DOS does not support such operations. The general rule is: do not call any functions that use the screen, read keyboard input or any file I/O routines; these should not be used in ISRs.

The same problem of re-entrancy also exists for many floating-point emulators. This effectively means that you should also avoid floating point mathematical operations in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you use. Even, if you are writing your ISR in Assembly language, DOS and many floating point emulators are not re-entrant. Of course there are ways to avoid this problem, such as those which activate when your ISR is called. Such solutions are, however, beyond the scope of this manual.

The second major concern when writing ISR's is to make them as short as possible in term of execution time. Spending long times in interrupt service routines may mean that other important interrupts are not serviced. Also, if you spend too long in your ISR, it may be called again before you have exited. This will lead to your computer hanging up and will require a reboot.

Your ISR should have the following structure:

- Push any processor registers used in your ISR.
- Put the body of your routine here
- Clear the interrupt bit by reading EFM104HR RXD register
- Issue the EOI command to the 8259 by writing 20h to 20h
- Pop all registers. Most C compilers do this automatically

The following C example shows what the shell of your ISR should be like:

```
void interrupt far new_IRQ_handler(void)
{

        IRQ_flag = 1;          // Indicate to process interrupt has occurred

        {
                               // Your program code to read UART
                               // read to a data buffer for example:
        Guc_buffer[Gi_bufpos++] = inp(gi_SERIAL_DATA);
        }

        outp(0x20, 0x20);      // Acknowledge the interrupt controller
}
```

Saving the Startup Interrupt Mask Register (IMR) and interrupt vector

The next step after writing the ISR is to save the startup-state of the interrupt mask register, (IMR) and the original interrupt vector you are using. The IMR is located in address 21h. The interrupt vector you will be using is located in the interrupt vector table which is an array of pointers (addresses) and it is locate din the first 1024 bytes of the memory (Segment 0 offset 0). You can read this value directly, but it is better practice to use DOS function 35h (get interrupt vector) to do this. Most C compilers have a special function available for doing this. The vectors for the hardware interrupts on the XT - bus are vectors 8-15, where IRQ0 uses vector 8 and IRQ7 uses vector 15. Thus if your EFM104HR is using IRQ5 it corresponds to vector number 13.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read the current IMR at I/O port 21h, and set the bit that corresponds to the IRQ. The IMR is arranged so that bit 0 is for IRQ0 and bit 7 is for IRQ7. See the paragraph entitled *Interrupt Mask Register (IMR*) earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h.

With the startup IMR saved and the interrupts temporarily disabled, you can assign the interrupt vector to point to your ISR. Again you can overwrite the appropriate entry in the vector table with a direct memory write, but this is not recommended. Instead use the DOS function 25h (Set Interrupt Vector) or, if your compiler provides it, the library routine for setting up interrupt vectors. Remember that interrupt vector 8 corresponds to IRQ0, vector 9 for IRQ1 etc.

If you need to program the source of your interrupts, do that next. For example, if you are using transmitted or received messages as an interrupt source program it to do that. Finally, clear the mask bit for your IRQ in the IMR. This will enable your IRQ.

Common Interrupt mistakes

Remember hardware interrupts are from 8-15, XT IRQ's are numbered 0-7. Do not forget to clear the IRQ mask bit in the IMR Forgetting to send the EOI command after ISR code. Disables further interrupts.

Example on Interrupt vector table setup in C-code:

```
void far _interrupt new_IRQ1_handler(void );            /* ISR function */
#define IRQ1_VECTOR     3                               /* Name for IRQ */
void (interrupt far *old_IRQ1_dispatcher)
        (es,ds,di,si,bp,sp,bx,dx,cx,ax,ip,cs,flags);    /* Variable to store
old IRQ_Vector */
void far _interrupt new_IRQ1_handler(void );

/*---------------------------------------------------------------------
| Function:     init_irq_handlers
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Set the pointers in the interrupt table to point to
|               our functions i.e. setup for ISR's.
|---------------------------------------------------------------------*/
void init_irq_handlers(void)
{
        _disable();
                old_IRQ1_handler = _dos_getvect(IRQ1_VECTOR + 8);
                _dos_setvect(IRQ1_VECTOR + 8, new_IRQ1_handler);
                Gi_old_mask = inp(0x21);
                outp(0x21,Gi_old_mask & ~(1 << IRQ1_VECTOR));
        _enable();
}


|/*---------------------------------------------------------------------
| Function:     restore, do this before exiting program
| Inputs:       Nothing
| Returns:      Nothing
| Purpose:      Restore the interrupt vector table.
|---------------------------------------------------------------------*/
void restore(void)
{
        /* Restore the old vectors */
        _disable();

                _dos_setvect(IRQ1_VECTOR + 8, old_IRQ1_handler);
                outp(0x21,Gi_old_mask);

        _enable();
}
```

Chapter 6 - EFM104HR SPECIFICATIONS

Host interface

16-bit PC/104 bus, XT-bus used for data

Modem specifications

Operational

Client-to-Server	V.90 or K56 flex 56Kbps download speed, upload speed 33,6Kbps via enhanced V.34
Client-to-Client	33600, 31200, 28800, 2400, 21600, 19200, 16800, 14400, 1200, 9600, 7200, 4800, 2400, 1200, 0-300bps
Fax data rates	14400, 1200, 9600, 7200, 4800, 2400, 1200, 300bps
Data format	Serial, binary, asynchronous
Modem compatibility	ITU V.90, K56flex
Status indicator	4 LED's

UART and I/O

UART compatibility	16C550
Oscillator frequency	1.8432MHz
Connection	Full hardware handshaking supported
Base addresses	32+4
Interrupts	2, 5,7,10,11,12,14 and 15
Digital I/O	8 TTL outputs, 8 TTL inputs w. 10K pd.

EFM104HR Electromechanical

Operating temperature range	-40 to +85C, Convection cooling
Humidity	RH up to 95% non-condensing
Altitude	-1000 to 30.000 ft
Vibration	Survival 10G peak
Power consumption	0.8W min; 1.2W normal

Chapter 7 - LIMITED WARRANTY

Real Time Devices, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from REAL TIME DEVICES. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, REAL TIME DEVICES will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to REAL TIME DEVICES. All replaced parts and products become the property of REAL TIME DEVICES. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by REAL TIME DEVICES, "acts of God" or other contingencies beyond the control of REAL TIME DEVICES), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN REAL TIME DEVICES. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND REAL TIME DEVICES EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL REAL TIME DEVICES BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS, WHICH VARY FROM STATE TO STATE.